Создание генератора кода для реализации доступа к закрытым данным класса

Автор: Абрамов М. С., учащийся МАОУ «Гимназия №13» г. Пензы

Руководители: Адамский С. С., Зайцев В. А.,учителя информатики МБОУ СОШ №30 г. Пензы.

Пенза, 2019 год

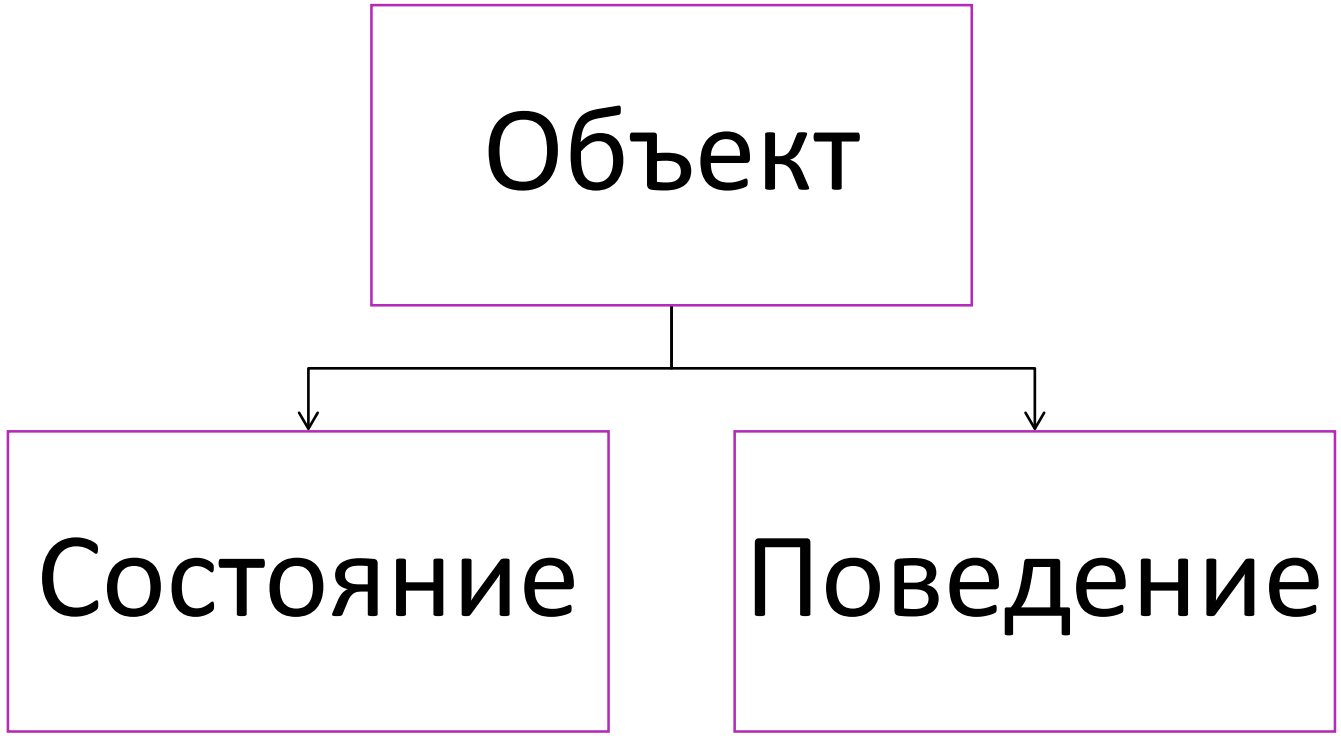
Цель работы

Оптимизация процесса написания вспомогательного PHP-кода в объектно-ориентированном программировании (ООП)

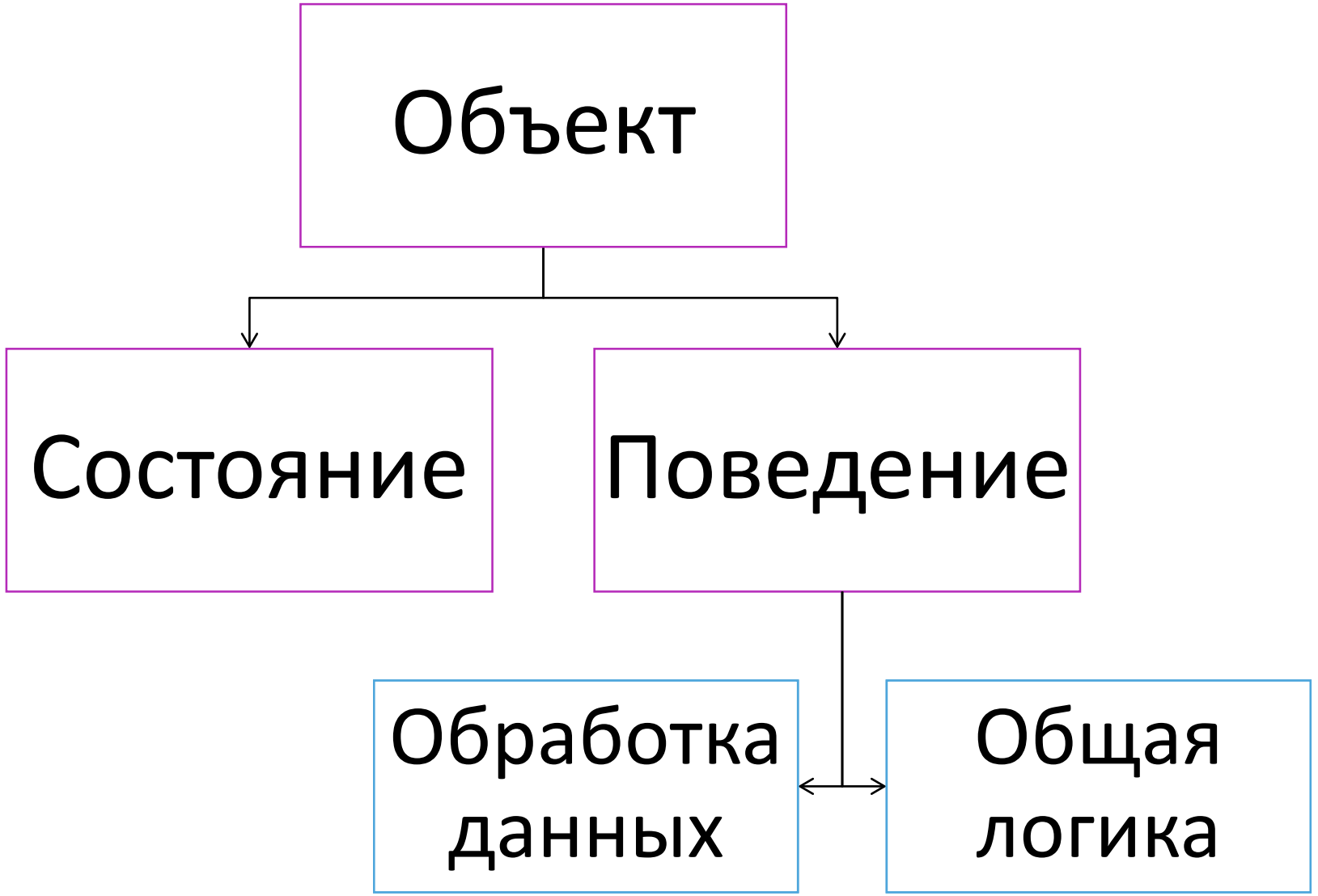
Задачи работы

- Выявить проблемы использования методов “get” и “set” в объектно-ориентированном программировании.
- Проанализировать информационные источники по выбранной теме.
- Выбрать подходящий способ решения выявленных проблем.
- Спроектировать библиотеку для автоматической генерации PHP-кода.
- Реализовать библиотеку на практике.

Структура объекта в ООП



Структура объекта в ООП



Методы обработки данных объекта

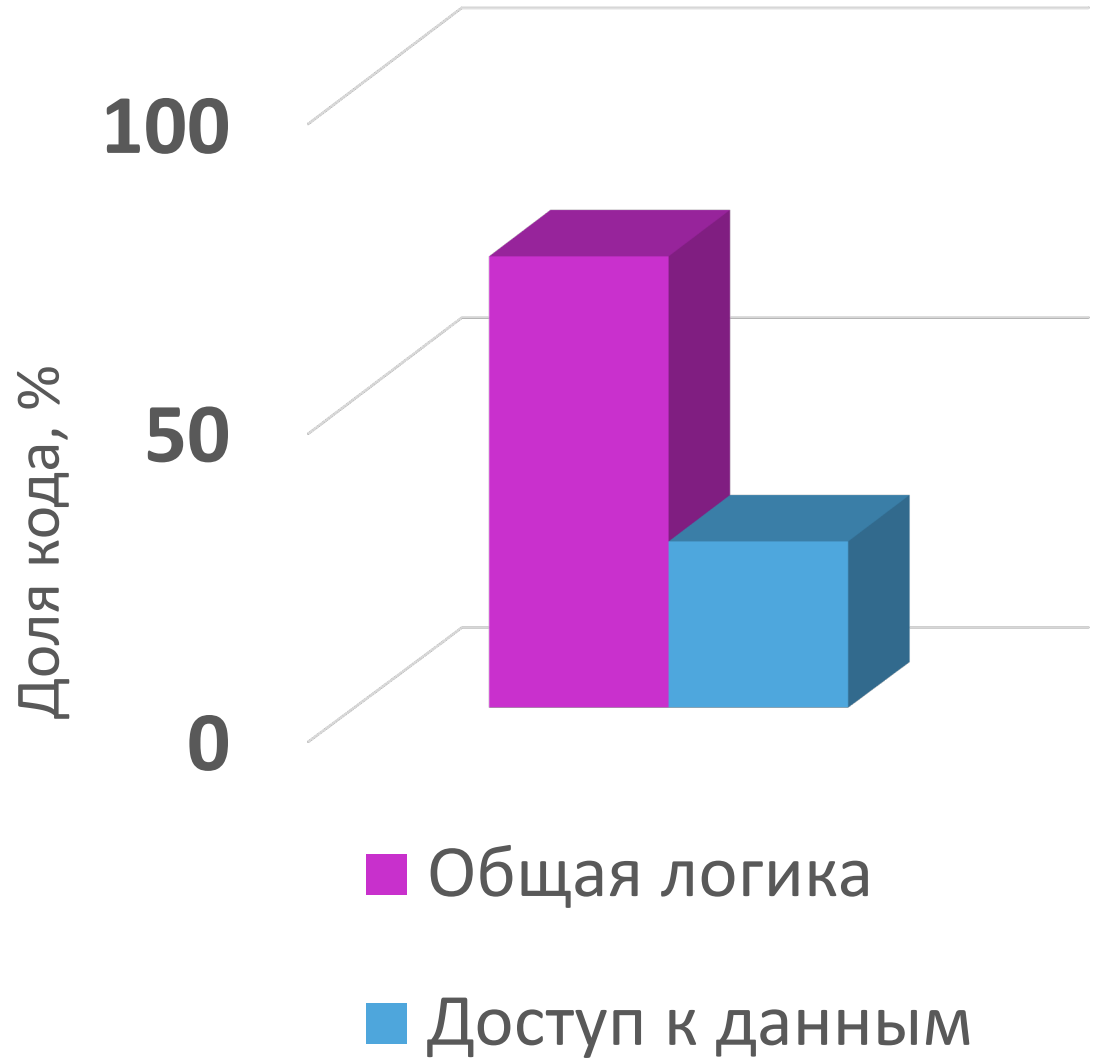
ПОЛУЧЕНИЕ

```
function get(): int
{
    return $this->field;
}
```

УСТАНОВКА

```
function set(int $val): void
{
    $this->field = $val;
}
```

Анализ тестового проекта “OfficeModel”



Модель офиса

- Person
- Employee
- PastPosition
- IdCard
- Room
- Unit
- Department
- Menu

Пример: класс цвета RGB (без библиотеки)

```
class Color {

    private $red;     private $green;    private $blue;
    public function __construct(float $red, float $green, float $blue) {…}

    public function equals(Color $another): bool {…}
    public function mix(Color $another): Color {…}
    private function avg(float $a, float $b): float {…}

    public function getRed(): float {…}
    public function setRed(float $value): void {…}
    public function getGreen(): float {…}
    public function setGreen(float $value): void {…}
    public function getBlue(): float {…}
    public function setBlue(float $value): void {…}
    private function validateColor(float $color): void {…}
}
```

Общая логика

Доступ к данным

Методы доступа для красного цвета

```
function getRed(): float {
    return $this->red;
}

function setRed(float $red): void {
    $this->validateColor($red);
    $this->red = $red;
}
```

Методы доступа для зелёного цвета

```
function getGreen(): float {
    return $this->green;
}

function setGreen(float $green): void {
    $this->validateColor($green);
    $this->green = $green;
}
```

Методы доступа для синего цвета

```
function getBlue(): float {
    return $this->blue;
}

function setBlue(float $blue): void {
    $this->validateColor($blue);
    $this->blue = $blue;
}
```

Выделен паттерн (шаблон) для метода доступа

```
public/protected/private function set/get Field(type $val) {
    if (<conditions>) {
        <handlers>
        <callbacks>
        $this->field = $val;
    } else {
        throw new Exception(description);
    }
}
```

Пример: класс цвета RGB (без библиотеки)

```
class Color {

    private $red;     private $green;    private $blue;
    public function __construct(float $red, float $green, float $blue) {…}

    public function equals(Color $another): bool {…}
    public function mix(Color $another): Color {…}
    private function avg(float $a, float $b): float {…}

    public function getRed(): float {…}
    public function setRed(float $value): void {…}
    public function getGreen(): float {…}
    public function setGreen(float $value): void {…}
    public function getBlue(): float {…}
    public function setBlue(float $value): void {…}
    private function validateColor(float $color): void {…}
}
```

Общая логика

Доступ к данным

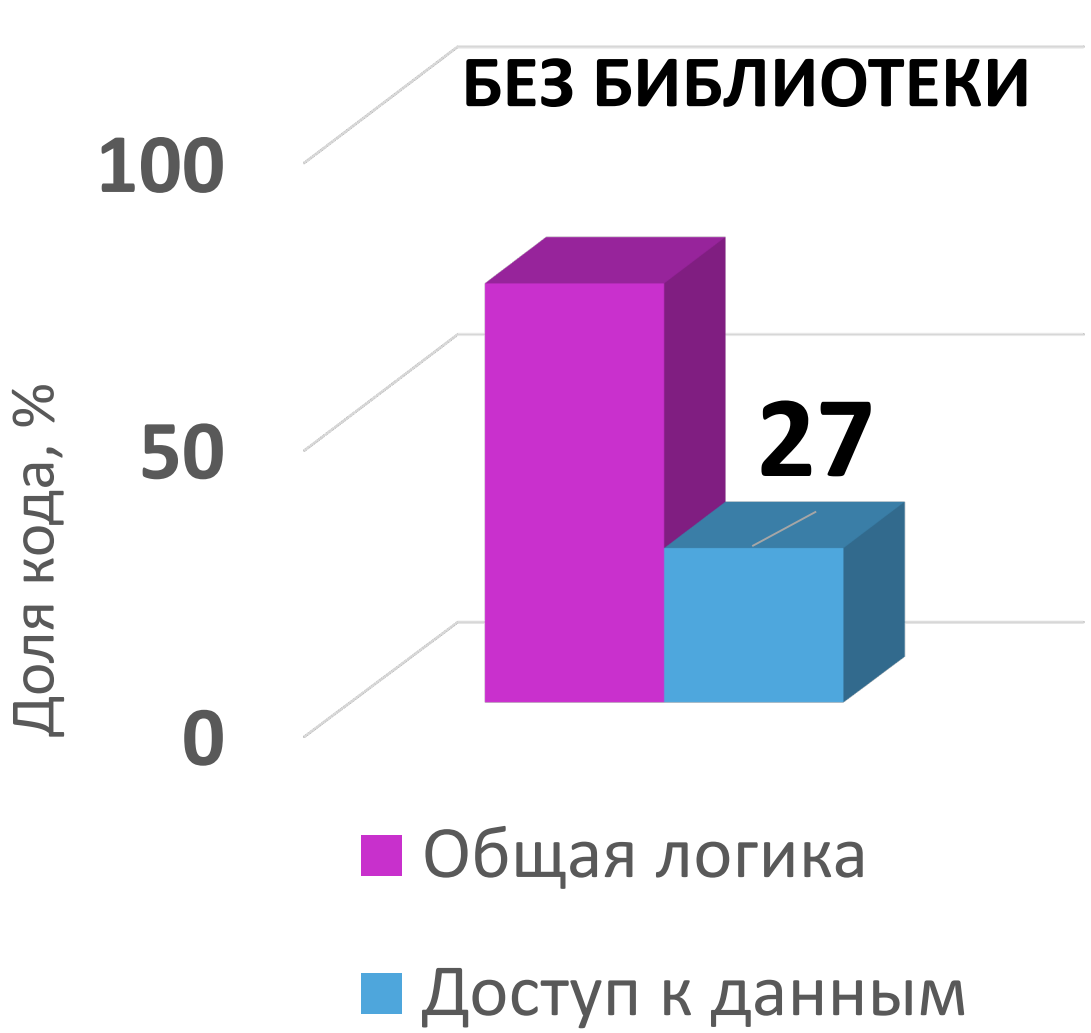
Пример: класс цвета RGB (с библиотекой)

```
class Color {
    use Axessors;
    private $red, $green, $blue; #: +axs int 0..255
    public function __construct(float $red, float $green, float $blue) {…}

    public function equals(Color $another): bool {…}
    public function mix(Color $another): Color {…}
    private function avg(float $a, float $b): float {…}

    ГЕНЕРИРУЕТСЯ ВО ВРЕМЯ ИСПОЛНЕНИЯ

}
```

Общая логика

Доступ к данным

Сравнительный анализ тестового проекта «OfficeModel»



Пример: класс «Person»

```php
class Person {
    private $name;
    private $surname;

    public function __construct(...) {
        $this->name = $name;
        $this->surname = $surname;
    }

    public function getName(): string {...}
    public function setName(...): void {...}
    public function getSurname(): string {...}
    public function setSurname(...) {...}
}
```

```php
class Person {
    use Axessors;
    private $name, $surname; #: +axs string

    public function __construct(...) {
        $this->name = $name;
        $this->surname = $surname;
    }

}
```

Синтаксис. Использование библиотеки

private $name, $surname; #: +axs string

```
public function getName(): string {
    return $this->name;
}
public function setName(string $name): void {
    $this->name = $name;
}
public function getSurname(): string {
    return $this->surname;
}
public function setSurname(string $surname): void {
     $this->surname = $surname;
}
```

Синтаксис. Модификатор доступа

private $name, $surname; #: +axs string

```
public function getName(): string {
    return $this->name;
}
public function setName(string $name): void {
    $this->name = $name;
}
public function getSurname(): string {
    return $this->surname;
}
public function setSurname(string $surname): void {
     $this->surname = $surname;
}
```

Синтаксис. Тип генерируемого метода

private $name, $surname; #: +**axs** string

```
public function getName(): string {
    return $this->name;
}
public function setName(string $name): void {
    $this->name = $name;
}
public function getSurname(): string {
    return $this->surname;
}
public function setSurname(string $surname): void {
     $this->surname = $surname;
}
```

Синтаксис. Объявление типа данных

private $name, $surname; #: +axs **string**

```
public function getName(): string {
    return $this->name;
}
public function setName(string $name): void {
    $this->name = $name;
}
public function getSurname(): string {
    return $this->surname;
}
public function setSurname(string $surname): void {
     $this->surname = $surname;
}
```

Варианты значений лексем комментария

+ *(public)*	rdb *(геттер)*	*int*
~ *(protected)*	wrt *(сеттер)*	*float*
- *(private)*	axs *(оба метода)*	*bool*
		string
		array
		object
		resource
		callable

x72

Паттерн для метода доступа

```
public/protected/private function set/get Field(type $val) {
    if (<conditions>) {
        <handlers>
        <callbacks>
        $this->field = $val;
    } else {
        throw new Exception(description);
    }
}
```

Паттерн для метода доступа

```
public/protected/private function set/get Field(type $val) {
    if (<conditions>) {
        <handlers>
        <callbacks>
        $this->field = $val;
    } else {
        throw new Exception(description);
    }
}
```

Паттерн для метода доступа

```
public/protected/private function set/get Field(type $val) {
    if (<conditions>) {
        <handlers>
        <callbacks>
        $this->field = $val;
    } else {
        throw new Exception(description);
    }
}
```

Пример: класс «Room»

```
class Room {

    private $number;

    public function __construct(...) {
        $this->number = $number;
    }
    public function getNumber(): int {...}
    public function setNumber(int $number)
{...}
    private function validateNumber(): void
{...}
}
```

#: +axs int > 0

```
class Room {
    use Axessors;
    private $number; #: +axs int > 0

    public function __construct(...) {
        $this->number = $number;
    }

}
```

Пример: класс «Room»

```
class Room {

    private $number;

    public function __construct(...) {
        $this->number = $number;
    }
    public function getNumber(): int {...}
    public function setNumber(int $number)
{...}
    private function validateNumber(): void
{...}
}
```

```
class Room {
    use Axessors;
    private $number; #: +axs int > 0

    public function __construct(...) {
        $this->number = $number;
    }

}
```

#: +axs int > 0

Синтаксис. Условия

+axs int ***> 0***

```
private function validateNumber(int $var): bool {
    if(!($var > 0)) {
        throw new Exception(…);
    }
}
```

Синтаксис. Пользовательские условия

+axs int `*$var < 0 & $var != -90*`

```
private function validateNumber(int $var): bool {
    if(!(*$var < 0 & $var != -90*)) {
        throw new Exception(…);
    }
}
```

Пример: класс «IdCard»

```
class IdCard {
    use Axessors;

    private $number; #: +axs int
    private $dateExpire;  #: +wrt int +rdb -> `$var = date('d.m.Y', $var)`

    /*...*/
}
```

```
class IdCard {
    private $number;
    private $dateExpire;
    public function __construct(int $number) {...}
    public function getNumber(): int {...}
    public function setNumber(int $number) {...}
    public function getExpirationDate(): string {...}
    public function setExpirationDate(int $ts) {...}
}
```

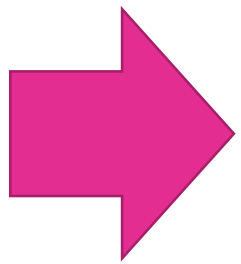
Синтаксис. Пользовательские обработчики данных

#: +wrt int +rdb -> `**$var = date('d.m.Y',$var)**`

Синтаксис. Обработчики данных

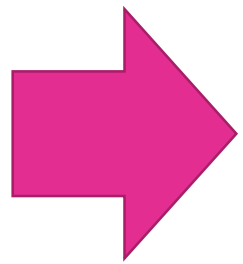
```
#: +wrt int +rdb -> inc
#: +wrt int +rdb -> dec
```

```php
class Departament implements Unit
{
    private $name;
    private $employees;

    public function __construct(string $name)
    {
        $this->name = $name;
    }

    public function getPersonCount(): int
    {
        return count($this->employees);
    }

    public function getName(): string
    {
        return $this->name;
    }

    public function setName(string $name): void
    {
        $this->name = $name;
    }

    public function getEmployees(): array
    {
        return $this->employees;
    }

    public function addEmployee(Employee $employee): void
    {
        $this->employees[] = $employee;
        $employee->setDepartament($this);
    }

    public function deleteEmployee(Employee $employeeToDelete): void
    {
        foreach ($this->employees as $key => $employee) {
            if ($employee === $employeeToDelete) {
                unset($this->employees[$key]);
            }
        }
    }
}
```



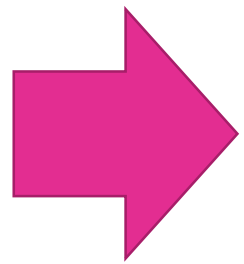
```php
class Departament implements Unit
{
    use Axessors;

    private $name; #: +axs string
    private $employees; #: +axs Array[Employee] => employee

    public function __construct(string $name)
    {
        $this->name = $name;
    }
}
```

```php
class Man
{
    private $name;
    private $surname;

    public function __construct(string $name, string $surname)
    {
        $this->name = $name;
        $this->surname = $surname;
    }

    public function getName(): string
    {
        return $this->name;
    }

    public function setName(string $name): void
    {
        $this->name = $name;
    }

    public function getSurname(): string
    {
        return $this->surname;
    }

    public function setSurname(string $surname): void
    {
        $this->surname = $surname;
    }
}
```



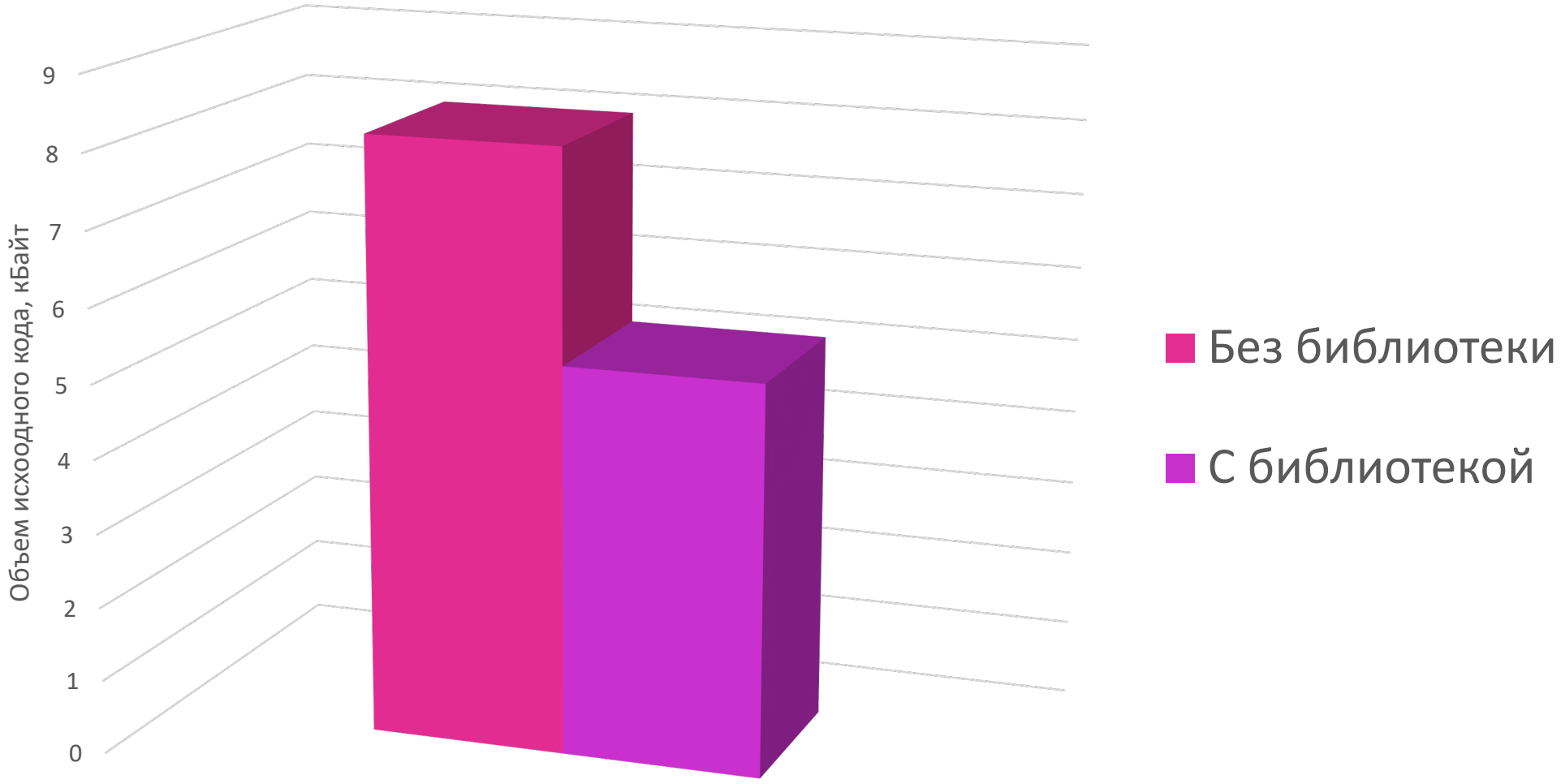
```php
class Man
{
    use Axessors;

    private $name, $surname; #: +axs string

    public function __construct(string $name, string $surname)
    {
        $this->name = $name;
        $this->surname = $surname;
    }
}
```

```php
class Room
{
    private $number;

    public function __construct(int $number)
    {
        $this->number = $number;
    }

    public function getNumber(): int
    {
        return $this->number;
    }

    public function setNumber(int $number): void
    {
        $this->number = $number;
    }
}
```



```php
class Room
{
    use Axessors;

    private $number; #: +axs int > 0

    public function __construct(int $number)
    {
        $this->number = $number;
    }
}
```

Динамика объема исходного кода тестового проекта



Принцип работы библиотеки

- получение списка классов, использующих библиотеку;
- получение информации о файлах с исходным кодом;
- чтение и обработка комментариев;
- сохранение полученной информации;
- генерация запрошенных методов;
- компиляция сгенерированных методов в байт-код PHP;
- встраивание скомпилированных методов в пользовательские классы.

Вывод

- В процессе анализа существующих способов уменьшения общего объёма однотипного кода был найден наиболее приемлемый способ решения проблемы.
- Были определены необходимые требования к проекту и средства его реализации.
- Был разработан наиболее короткий синтаксис описания методов доступа.
- Библиотека была спроектирована и реализована на практике.

Ссылки на исходный код

https://www.github.com/NoOne4rever/Axessors

README.md

Axessors

build passing | Scrutinizer 9.73 | coverage 97 % | license GPL

Generator of getters and setters for PHP.

Installation

`composer require noone4rever/axessors` or just download AxessorsPHAR.

You can see this package on packagist.

System requirements

You need to use PHP 7.1 or newer to run code with Axessors.